AD-A211 758

(2)

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| | | |
|---|---|---|
| **REPORT DOCUMENTATION PAGE** | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
| 1. REPORT NUMBER SCA 0001 | 2. GOVT ACCESSION NO. | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle) Building a Software Package for Solving Large Sparse Nonlinear Systems of Equations | | 5. TYPE OF REPORT & PERIOD COVERED Final Report 85 July 01- 85 Dec. 31 |
| | | 6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s) Scientific Computing Associates, Inc. | | 8. CONTRACT OR GRANT NUMBER(s) F49620-85-C-0082 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS N/A |
| 11. CONTROLLING OFFICE NAME AND ADDRESS USAF AFSC AFOSR Building 410 Bolling AFB Washington, D.C. 20332-6448 | | 12. REPORT DATE 24 Feb. , 1986 |
| | | 13. NUMBER OF PAGES 20 |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) USAF, AFSC | | 15. SECURITY CLASS. (of this report) unclassified |
| | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

unrestricted.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

DTIC ELECTE AUG 17 1989 S B D

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Nonlinear equations, sparse equations, Newton's method, Krylov subspace methods, Jacobian free methods, Nonlinear preconditionings, Discrete Newton's method, inexact Newton's method.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

We developed a preliminary version of a general purpose code based on a nonlinear version of GMRES, a method for solving nonsymmetric linear systems. Using the Jacobian-free approach, we were able to solve relatively hard problems in a short period of time and with a minimum of programming effort, thus demonstrating the viability of the proposed code. Among the strongest points of the projected package are its ease of use and the variety of proposed techniques. → over (cont)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

89 8 17 123

Air Force Small Business Innovation Research Program

Topic 199: Research in Mathematics of Control and Dynamics for Aerospace Systems

**Final Report on Phase I Research**

Report # SCA0001

# Building a Software Package for Solving Large Sparse Nonlinear Systems of Equations

Contract # F49620-85-C-0082

*Scientific Computing Associates, Inc.*
246 Church Street
Suite 307
New Haven, CT 06510

(203) 777-7442

(1)

## Summary

## 1. Research Objectives

Although there has recently been progress in research on nonlinear problems, there is no commercially available software package for dealing with them. The objective of Phase I in this project was to write and test a preliminary code to show the feasibility of a general purpose nonlinear solver. In doing so, we concentrated on:

1) Considering appropriate simple techniques for handling nonlinearities;

2) Choosing the most appropriate linear system packages to incorporate in the package;

3) Improving and incorporating more sophisticated algorithms for handling nonlinearities.

We emphasized the first and third objectives. For the second objective we have also done some tests with PCGPAK, the Preconditioned Conjugate Gradient Package of subroutines for the iterative solution of large, sparse, nonsymmetric systems of linear equations, a package implemented by Scientific Computing Associates.

We have developed a preliminary version of a general purpose code based on a nonlinear version of GMRES, a method for solving nonsymmetric *linear* systems ~~[15]~~. While studying the feasibility of a general purpose nonlinear solver, we decided that such a package should be more general than that which we originally proposed. That is, such a package should not only solve nonlinear equations, but also should be able to accelerate already existing packages. The reason is that there are a host of specialized codes that have taken years to develop by engineers and scientists; it seems inefficient to replace these codes entirely. Our general purpose nonlinear solver can take the outer loops of these codes and accelerate them. We explain this promising approach in Section 2.2.

Using the Jacobian-free approach, we were able to solve relatively hard problems in a short period of time and with a minimum of programming effort. As we stressed in our initial proposal, when combined with a method like GMRES this approach has the enormous advantage of being easy to implement because it requires no manipulation of the Jacobian matrix in any form. Yet it is just as powerful as Newton's method since it is in effect an inexact Newton method. This makes it possible to develop a general purpose accelerator. For example, consider an iteration of the form $u_{n+1} = G(u_n)$. The purpose of this iterative scheme is to attempt to solve the equation

$$u - G(u) \equiv F(u) = 0.$$

Solving the above equation by a Newton-type iteration would be in most cases impossible for even a slightly complicated function $G$. On the other hand NLGMR, the nonlinear GMRES, requires no explicit Jacobian; it requires only the subroutine that computes $G(u)$ for a given $u$.

## 2. Status of the Research Effort

### 2.1. NLPACK: a general purpose nonlinear system solver

NLPACK, our projected package, has two main branches. The first is the Jacobian-free part of the package, which contains the general purpose accelerator. The second is the Newton-GMRES package using explicit or internally computed Jacobians. In Phase I we have concentrated on the first branch because it constitutes the more original and important work. Both branches however, will utilize the same NLGMR subroutine as a nonlinear solver, which is described next.

#### 2.1.1. NLGMR: A nonlinear version of GMRES

NLGMR is a nonlinear extension of GMRES, a method devised for solving linear equations. It can be viewed as Newton's method using the linear GMRES for solving the Jacobian system. This constitutes the kernel of NLPACK.

In what follows we are interested in solving the nonlinear system

$$F(u) = 0, \tag{2.1}$$

where $F$ is a nonlinear application from $\mathbf{R}^N$ to $\mathbf{R}^N$. Throughout we denote by $\|.\|$ the Euclidean norm in $\mathbf{R}^N$, and $J_F(u)$ the Jacobian matrix of the application $F$ at the point $u$. When there is no ambiguity, $J_F(u)$ will be denoted by $J_F$.

Suppose that $u_n$ is the current approximation and that we wish to find a new approximation of the form $u_{n+1} = u_n + \delta$. In the nonlinear version of GMRES [18], the vector $\delta$ is written in the form

$$\delta = \sum_{i=1}^{m} \alpha_i v_i, \tag{2.2}$$

where the $v_i$'s are $m$ orthonormal vectors, the determination of which will be clarified shortly, and the $\alpha$'s are unknowns to be determined. Ideally we would want to minimize

$$\|F(u_n + \delta)\|$$

over all vectors $\delta$ to get the new iterate $u_{n+1} = u_n + \delta$. Unfortunately, this would lead to a numerically difficult minimization problem. Instead, we can linearize $F$ about the point $u_n$ by writing $F(u_n + \delta) \approx F(u_n) + J_F\delta$ where $J_F$ represents the Jacobian of $F$ at $u_n$, and we minimize

$$\|F(u_n) + J_F\delta\| \tag{2.3}$$

(1) ***Start:*** Choose $u_0$ and a dimension $m$ of the Krylov subspace. Set $n = 0$.

(2) ***Arnoldi process:***

- Compute $\beta = \|F(u_n)\|$ and $v_1 = F(u_n)/\beta$.
- For $j = 1, 2, .., m$ do:

$$h_{i,j} = (J_F v_j, v_i), \quad i = 1, 2, ..., j,$$
$$\hat{v}_{j+1} = J_F v_j - \sum_{i=1}^{j} h_{i,j} v_i,$$
$$h_{j+1,j} = \|\hat{v}_{j+1}\|, \quad \text{and}$$
$$v_{j+1} = \hat{v}_{j+1}/h_{j+1,j}.$$

Define $H_m$ to be the $(m+1)$ x $m$ (Hessenberg) matrix whose nonzero entries are the coefficients $h_{ij}$, $1 \leq i \leq j+1$, $1 \leq j \leq m$ and define $V_m \equiv [v_1, v_2, \ldots v_m]$

(3) ***Form the approximate solution:***

- Find the vector $y_m$ that minimizes the function $\gamma(y) \equiv \|\beta e_1 - H_m y\|$, where $e_1 = [1, 0, \ldots 0]^T$, among all vectors $y$ of $R^m$.
- Compute $\delta_n = V_m y_m$ and $u_{n+1} = u_n + \delta_n$.

(4) ***Restart:*** If satisfied stop, else set $u_n \leftarrow u_{n+1}$ , $n \leftarrow n+1$, and goto (2).

**Algorithm 1:** NonLinear Generalized Minimal Residual (NL-GMR)

over all vectors $\delta$ of the form (2.2). The above minimization can be achieved by performing $m$ steps of the GMRES algorithm [15], applied to the linear equation

$$J_F \delta = -F(u_n), \tag{2.4}$$

starting GMRES with the initial solution $\delta^{(0)} = 0$. Notice that solving this equation exactly will yield the Newton direction $J_F^{-1} F(u_n)$. Initially, we assume that the Jacobian $J_F$ is available and will temporarily not concern ourselves with the way in which it is computed.

Algorithm 1 is a restarted version of the GMRES algorithm which at every outer iteration generates the orthonormal vectors $v_i, i = 1, 2, \ldots m-1$, and then builds the vector $\delta_n$ that minimizes $\|F(u_n) + J_F \delta\|$. In GMRES the $v_i$'s are computed such that they form an orthonormal basis of the Krylov subspace *span* $\{v_1, J_F v_1, \ldots, J_F^{m-1} v_1$ where $v_1$ is obtained by normalizing $F(u_n)$.

Steps 2 and 3 of Algorithm 1 are precisely the GMRES(m) process for solving the linear system $J_F\delta = -F(u_n)$, with the particular initial guess $\delta^{(0)} = 0$ (see [15]). With the standard notation $r_0$ for the initial residual of the above linear system (here $r_0 = F(u_n)$) each outer loop of the above algorithm, consisting of steps 2, 3, and 4, is divided in two main stages. The first stage is an Arnoldi loop that builds an orthonormal basis $V_m = [v_1, v_2, \ldots, v_m]$ of the Krylov subspace $K_m \equiv span\{r_0, Jr_0, J^2r_0, \ldots, J^{m-1}r_0\}$. If we denote by $V_j$ the $N$ x $j$ matrix with column vectors $v_1, v_2, \ldots v_j$, and by $\bar{H}_m$ the $(m+1)$ x $m$ upper Hessenberg matrix whose nonzero entries are the $h_{i,j}, 0 \leq i \leq j+1,\ 1 \leq j \leq m$, then it is easy to establish the following relation [15] which is crucial in the development of GMRES,

$$J_F V_m = V_{m+1}\bar{H}_m. \tag{2.5}$$

Step (3) of NLGMR computes in the subspace $K_m$ the approximate solution $\delta_n$ that minimizes the residual norm $||J_F\delta + F(u_n)||$. Writing $\delta = V_m y$, where $y$ is in an $m$-vector, and observing that by definition of $v_1$ we have $F(u_n) = \beta v_1$ we see that $y$ must minimize

$$||V_m y + \beta v_1|| = ||V_{m+1}[\bar{H}_m y - \beta e_1]||.$$

This explains the least squares problem of size $m+1$, with upper Hessenberg matrix $\bar{H}_m$ in step (3) of the algorithm. Then $u_{n+1}$ is computed at the end of step (3), by adding $\delta_n$ the optimal $\delta$ found. However, when damping is included this will be replaced by

$$u_{n+1} = u_n + \alpha_n\delta_n$$

where the damping coefficient will be computed by some damping subroutine, to be called from NLGMR.

For simplicity, we have omitted several details on the practical implementation in the above presentation, which are discussed at length in [15]. For example, in practice one computes progressively the least squares solution $y_m$ in the successive steps $j = 1, \ldots, m$ of stage 2. Thus, at every step, after this least squares solution is updated, we obtain the residual norm of the corresponding approximate solution $x_k$ without having to actually compute it (see [15]). This allows us to stop at the appropriate step.

The GMRES algorithm is theoretically equivalent to GCR [5] and to ORTHODIR [10] but is less costly in terms of storage and arithmetic [15]. For a synthesis and general description of available conjugate gradient type methods see [14]. A comparison of the cost of each step of these algorithms shows that for large enough $m$, GMRES costs about 1/3 less than GCR/ORTHOMIN in arithmetic while storage is roughly divided by a factor of two. Another appealing property of the algorithm is that in exact arithmetic, the method does not break down or, to be more accurate, that it breaks down only when it delivers the exact solution [15].

Restarting the algorithm as is done in step (4) corresponds to taking a new Newton step. In fact if the GMRES algorithm solves the linear system $J_F\delta = -F(u_n)$ exactly, or rather with sufficient accuracy by taking, for example, $m$ sufficiently large, then it is clear that the above algorithm is nothing but Newton's method, in which the Jacobian linear systems are solved by GMRES. Thus, the above algorithm constitutes a basic Newton/GMRES method. It constitutes the kernel of the proposed package.

Perhaps one of the most important aspects of the above GMRES iteration is that the Jacobian matrix $J_F$ need not be explicitly available. The only operations on the Jacobian matrix $J_F$ that are required from Arnoldi's method are matrix-vector multiplications $w = J_F v$, which can be approximated by

$$J_F(u)v \approx \frac{F(u+hv) - F(u)}{h} \tag{2.6}$$

where $u$ is the point where the Jacobian is being evaluated and $h$ is some carefully chosen small scalar. The idea of exploiting the above approximation is not new and was extensively used in the context of ODE methods [4, 7, 3, 2], in eigenvalue calculations [6], and is quite common in nonlinear equation solution methods and optimization methods (see, for example, [9, 11, 18]).

#### 2.1.2. Using NLGMR for accelerating a basic iteration

There are numerous scientific codes in which the most time consuming loop of a program is spent performing an iteration of the form

$$u_{n+1} = M(u_n).$$

The above scheme can be any stationary iterative method such as nonlinear Gauss-Seidel, SSOR, line Jacobi, Multigrid sweeps, or other particular schemes. It is important to observe that this iteration is only attempting to solve the nonlinear equation $F(u) \equiv u - M(u) = 0$. Since NLGMR only requires function evaluations, it can be applied to solving $F(u) = 0$ with no particular difficulty by using the user-supplied subroutine performing one step of the above basic iteration.

In typical cases the basic iterative methods have been tested and fine-tuned for several years on a particular problem for which it is specifically designed. Such examples abound in engineering. We have considered two such applications from combustion simulations and fluid dynamics that are described in Section 2.3.

### 2.1.3. Techniques to improve global convergence properties

As is well known, the main disadvantage of Newton's method is that it only converges when the initial guess is close to the solution. There are basically two classes of methods for preventing divergence. The first is a combination of damping and trust regions techniques and the second is homotopy. Both methods should be implemented in a general purpose code. In our feasibility study, we implemented elementary versions of both approaches. However, we consider these implementations rudimentary; substantial effort should be spent in perfecting them for the proposed software package because they are among the most important factors of efficiency.

#### Damping

A traditional method for improving reliability in Newton's method is to replace the usual iteration by the so-called damped Newton iteration

$$u_{n+1} = u_n + \alpha_n \delta_n \tag{2.7}$$

where $\delta_n = -J_F^{-1} F(u_n)$ and where the scalar $\alpha_n$ is chosen, for example, to minimize $\|F(u_n + \alpha\delta_n)\|$ over $\alpha \in [0,1]$. This requires a line search and may be time consuming to compute. However, more practical mid-way alternatives can be used. A simple strategy is to start with $\alpha = 1$ and check if the corresponding $\|F(u_{n+1})\|$ shows a sufficient decrease as compared to $\|F(u_n)\|$. If this is not the case we halve $\alpha$ and repeat the test until a decrease of the norm is achieved. We implemented this strategy in our preliminary version of NLGMR. Under very mild conditions, there is always an $\alpha$ sufficiently small for which the norm decreases and, although global convergence is not guaranteed by the theory, it is likely to be achieved in practice [17, 12]. It is also easy to show that, even when the linear systems are solved only approximately by reducing the initial residual norm $\|F(u_n)\|$ by a ratio $<1$, it is possible to achieve a decrease in the norm $\|F(u_n + \alpha\delta_n)\|$ for $\alpha$ small enough.

#### Modified Homotopy

A simple but attractive form of improving global convergence properties of NLGMR is a technique borrowed from [18] that can be formulated as follows for the case where $F(u) = u - M(u)$. At step $n+1$ choose a parameter $\lambda$ and solve the following equation with respect to $u$ to get $u_{n+1}$:

$$G(u,\lambda) \equiv \lambda(u - M(u)) + (1-\lambda)(u - M(u_n)) = 0. \tag{2.8}$$

Observe that $G(u,1) = u - M(u)$. The principle of this damping method is to start with $\lambda_0 = 0$, for which the solution of (2.8) is trivial, and then drive $\lambda$ to 1 while solving $G(u,\lambda) = 0$ for each new value of $\lambda$ using the previous $u_n$ as the initial guess to the new system $G(u, \lambda) = 0$. This has proven a simple though effective technique for improving global convergence behavior. We have implemented several different strategies for varying $\lambda$. The most promising is to take a conservative approach at first and then increase $\lambda$ according to the decrease obtained in the residual norm $||u - M(u)||$ of two successive iterates. This ensures that super-linear convergence still holds at the limit.

We have put more effort in developing homotopy techniques than damping techniques.

## 2.2. Current Software

In this section we briefly describe the software developed during Phase I of the project. We briefly describe the package as it stands and illustrate its usage on a few benchmark problems. Before starting we would like to point out a few predominant features of the current package. First, we have maintained a high level of modularity. This is helpful not only to the prospective user but also to us when debugging and testing. Second, we have maintained generality and flexibility by making many of the features optional, to be turned on and off at will. For example, we may run the code without the homotopy process or with the more standard damping technique or a combination of both.

Finally, we should stress that we plan to build a package that can be used either as a nonlinear solver or as an accelerator of existing iterative processes. Thus, NLPACK should be able to accommodate and take advantage of a variety of the user-provided information. It should also easily interface with a wide range of different problems.

### 2.2.1. Jacobian-free part of NLPACK

As we indicated earlier, we were able to solve relatively hard problems in a short period of time and with a minimum of programming effort by using the Jacobian-free approach. We concentrated most of our effort in Phase I on this part of the package. A short description follows.

- *An easy to follow model driver model and corresponding data file.* At present this is where the user allocates workspace, chooses parameters and selects various options for damping and modified homotopy. Also selected are the dimension of the Krylov space, the parameters dealing with the repetition of the preconditioning, and different tolerances for stopping the iterations. The user also chooses the preconditioning by declaring different external subroutines, method, stopping routine, output routine, damping routine, homotopy routine and declares the user supplied function routine $M$ or $F$.
- *NLGMR.* This is the main subroutine of the Jacobian-free part of NLPACK. It is called from the driver and in turn calls the preconditioner, i.e. an iterative process to accelerate by GMRES

as described in Section 2.1.2. The user can supply his own preconditioner or use one of three supplied by the package.

- *Three general purpose preconditioners*: SSOR, nonlinear Jacobi, and functional iteration (which includes the case of no preconditioning). If the user chooses to use one of these, he must provide the function subroutine, which is called from these preconditioners. These are needed in a scalar form, i.e. the individual components are required.
- *Subroutines for damping, modified homotopy, stopping, output.* The user can replace any of these routines by his own.

### 2.2.2. Newton/PCGPAK part of NLPACK

We wrote a sample program to solve a function arising from a five point operator, using Newton's methods and the existing package PCGPAK. The user supplies both the function and the Jacobian matrix in either Yale Sparse Matrix format or in diagonal format. The user has the option of switching on damping in this program. This is only a preliminary version of the Newton code we plan to provide in this part of the package.

## 2.3. Numerical Results

It was important to test NLPACK on a variety of realistic problems — ranging from easy to hard — issued from real applications. We selected the following test problems.

**Problem number 1**

This problem is a simple nonlinear partial differential equation of an elliptic type, with a parameter $c$ that can be changed to vary the degree of nonlinearity in the equation:

$$-\Delta u + c\ u(u_x + u_y) = f$$

on the unit square, and $u = g$ on the boundary.

**Problem number 2**

This problem is similar to Problem 1, but it has an exponential-type nonlinearity:

$$\Delta u + c\ e^u = 0$$

on the unit square, and $u = 0$ on the boundary. Again the parameter $c$ can be varied to change the difficulty of the problem.

**Problem number 3: Incompressible viscous flow calculations**

The flow of an incompressible viscous fluid in a cavity driven by a uniformly moving boundary has been successfully calculated in our Phase I project for relatively high Reynolds numbers. The driven cavity problem has received a great deal of attention in the computational fluid dynamics literature as a test problem for comparing numerical techniques for solving the Navier-Stokes equations (see, for example, [1, 8, 16] and the references therein).

We have used the stream function/vorticity function form of the two-dimensional Navier-Stokes equation :

$$\Delta\omega - R[\Psi_y\omega_x - \Psi_x\omega_y] = 0$$

$$\Delta\Psi + \omega = 0,$$

on a rectangle, where $\Psi$ is the stream function, and $\omega$ is the vorticity. Boundary conditions were obtained by translating those obtained from the primitive equations which use the velocities and the pressure as unknowns. These conditions are

$$\Psi(x,0) = \Psi_y(x,0) = 0$$

$$\Psi(0,y) = \Psi_x(0,y) = 0$$

$$\Psi(1,y) = \Psi_x(1,y) = 0$$

$$\Psi(x,1) = 0$$

$$\Psi_y(x,1) = 1.$$

From our experience in Phase I, implementing a Jacobian method such as NLGMR/SSOR is much easier than a Newton-type technique using the Jacobian. This illustrates one of the main advantages of NLGMR: the programmer's task is simple because Jacobians are not needed and not manipulated. Our tests on the driven cavity problem have been a relatively simple task, compared to what we can expect from implementing a multigrid code, for example.

**Problem number 4 : Acceleration of the *TEACH* combustion code**

*TEACH* is a multidimensional steady-state compressible flow algorithm based on a low-order control-volume discretization of the primitive variable equations in conservation form over orthogonal, tensor-product grids [13]. In the applications for which it is best suited, a "lumped" chemical reaction of Arrhenius form is postulated, involving only a handful of species. The nonlinearities arise directly from the convective terms and the reaction source terms, and indirectly through temperature-, pressure-, and composition-dependent laminar transport properties and thermodynamic coefficients, and the velocity-depend ,ut turbulent transport properties. A multistage variation of the block Gauss-Seidel method is used to solve this nonlinear system. The outermost stage consists of cycling between a Poisson-like pressure correction equation (derived from a truncated substitution of the discrete momentum equations into the discrete continuity equation, which is thus eliminated) and the transport equations for all of the other unknown fields. Within this latter block, the equations are relaxed cyclically, field-by-field. Within the sub-blocks at the level of the individual fields, the updates are generally computed in a block-line fashion (although pointwise relaxation schemes have also been applied), so that an elemental tridiagonal matrix solver is the largest implicit aggregate in the overall calculation. In piactice, under-relaxation of the updates is necessary. Variations in the cyclic order, in the sequence of convergence tolerances imposed on intermediate iterates, and in the frequency of the updating of the nonlinear coefficients abound, depending upon the application.

A parallel project at Scientific Computing Associates, also funded by AFOSR, is to improve the *TEACH* code by incorporating state-of-the art nonlinear methods. NLGMR is one of the methods considered. Some preliminary tests have shown NLGMR to be marginally successful, i.e. savings of the order 20% to 28% have been realized.

### 2.3.1. Solution of the driven cavity problem

In Figures 1 and 2 we plotted the output from NLGMR applied to the driven cavity problem on a $16 \times 16$ grid on the unit square, with Reynolds number 100. These plots agree well with plots obtained by researchers using other methods [1, 8, 16]. NLGMR was able to solve the problem with higher Reynolds numbers and more grid points: our largest run was $R = 100$ and a mesh of size $64 \times 64$. The only place where NLGMR seems to need improvement is in its efficiency. For example, another way of solving this problem is to use the multigrid method, which turn out to be more efficient than our present version of NLGMR. On the other hand, the multigrid method requires a much greater coding effort and is less suitable for a general purpose package. In fact, we can use an existing multigrid code as a preconditioner for NLGMR, and we expect that NLGMR/multigrid will accelerate the multigrid solution of this problem for many grid points and large Reynolds numbers.

### 2.3.2. NLGMR as an accelerator

In Figure 3 we illustrate NLGMR as an accelerator. For this diagram we solved the driven cavity problem on a $16 \times 16$ mesh on the unit square to an accuracy of 1.0d-6. We made the problem increasingly difficult by increasing the Reynolds number from 1.0 to 600 and measured the efficiency of the method by counting the number of function calls. We compared SSOR with NLGMR applied to accelerate (and improve) SSOR. The results confirmed our beliefs about NLGMR as an accelerator, and provided data about NLGMR as an enhancer of other methods. Indeed, SSOR broke down at Reynolds numbers higher than 150. NLGMR/SSOR performed better than SSOR whenever SSOR solved the problem. In addition, NLGMR/SSOR solved problems which SSOR could not handle.

It is interesting to remark that for Reynolds numbers higher than 400 the convergence of NLGMR was becoming slow. We switched on some of the modified homotopy options, without much effect (although these turned out to greatly reduce costs on other problems). Then we tried slightly increasing the maximum dimension of the Krylov space and we switched on the damping option; these options sped up the convergence rate of NLGMR. This example illustrates the importance of having easy access to the different parameters.

**2.3.3. Some more tests on the usage of NLPACK**

In Figure 4 we illustrate the usage of both branches of NLPACK. We solved Problem 1 on a $20 \times 20$ grid on the unit square, with $c = 5.0$. We plotted the norm of the residual against CPU time and the number of function calls because function evaluations are inexpensive in this case. Indeed, for NLGMR the CPU time is almost proportional to the number of function calls, but for Newton/PCGPAK this is no longer true. Naturally Newton/PCGPAK using GMRES as an iterative method performs better because it takes advantage of the knowledge of the Jacobian and its structure. For the sake of comparison, we made the somewhat arbitrary decision to count each Jacobian call as $nz$ scalar function calls, where $nz$ is the number of non-zero entries in the Jacobian matrix.

We draw two conclusions from this experiment. First, the performance of NLGMR compares well with other methods (for some problems it might be well worth paying the extra time for not having to provide and store the Jacobian). Second, providing the Newton/PCGPAK part of the package is worthwhile because one might get substantial improvements if the Jacobian is available.

In Figure 5 we compare different preconditioners in solving problem number 2 for $c = 5.0$ on a $20 \times 20$ grid on the unit square. We compare the preconditioners currently available and a preconditioner from a user-supplied method to solve the problem $u_{n+1} = -c\Delta_h^{-1}(\exp(u_n))$. For this problem the user-supplied Picard-type method is very efficient. Indeed, it performs better than all the other preconditioners. We can also see that, as expected, SSOR does better than the other preconditioners. The Jacobi preconditioner does not improve over no preconditioning for this problem. Once again this emphasizes the importance of modularity in NLPACK.

## 2.3.4. Other numerical experiments and work in progress

We experimented with repeating the preconditioner (or the user method if that is the case). For some problems it pays replacing $M$ from $u_{n+1} = M(u_n)$ by $M^k$ for $k$ in the range 2 to 5. However, for larger values of $k$ the method deteriorates in that the total number of iterations in GMRES increases.

We also tested different strategies for modified homotopy. We concluded that homotopy is very useful for some problems. Moreover, it makes a difference how fast and when $\lambda$ is decreasing (right now we have five strategies for choosing $\lambda$). At present, modified homotopy has the drawback of being based on the user's intuitive feeling or heuristic thinking. For this reason we plan to implement a new homotopy technique that generalizes the current modified homotopy method. This will give the user the choice of switching on the automatic choice of $\lambda$ and/or restarting or choosing it according to his knowledge of the problem.

At the time of this writing, we continue the development of NLPACK on several fronts. We are almost finished implementing the homotopy in NLGMR as explained above. We are now working on turning the sample Newton/PCGPAK program into a working tool, and we have started providing a few routines to interface some standard formats for user-provided Jacobians with the Yale Sparse Matrix format (used by PCGPAK). Finally, we are expanding the pool of benchmark problems.

## 2.4. Conclusion

In a relatively short time we have been able to write a short version of our package, consisting mainly of the Jacobian-free part, and to use it to solve a few benchmark problems that demonstrate the viability of the proposed code. Perhaps among the strongest points of the projected package are its ease of use and the variety of proposed techniques.

It is important to realize that this is only preliminary work and that there are many issues that will be raised concerning the performance improvement of the method. Among them are at least two important issues that will require substantial programming effort and that will undoubtedly improve the effectiveness of the current NLGMR package

1. Scaling problems. Variables having widely different orders of magnitude are common. In these situations it is clearly important to rescale the variables according to size. This was stressed in particular by Brown and Hindmarsh [2] who propose a scaled version of IOM, a method

related to GMRES. Because the Jacobian is approximated by finite differences, a bad scaling can lead to poor performance.

2. Robust homotopy and damping techniques. As was mentioned earlier there are elaborate damping and homotopy methods that require programming efforts that go beyond a six month period. These include trust region techniques combined with steepest descent methods, and the more standard homotopy in place of the modified homotopy method implemented in our preliminary package.

## 3. Publications

None.

## 4. Personnel

1. Dr. Mark W. Angevine, Principal Investigator
2. Dr. Youcef Saad
3. Dr. Tony F.C. Chan
4. Dr. Peter Farkas

## 5. Interactions (Coupling Activities)

None.

## 6. New discoveries, inventions, or patent disclosures and specific applications stemming from the research

We have proven the feasibility of our proposed software package for solving general nonlinear equations.

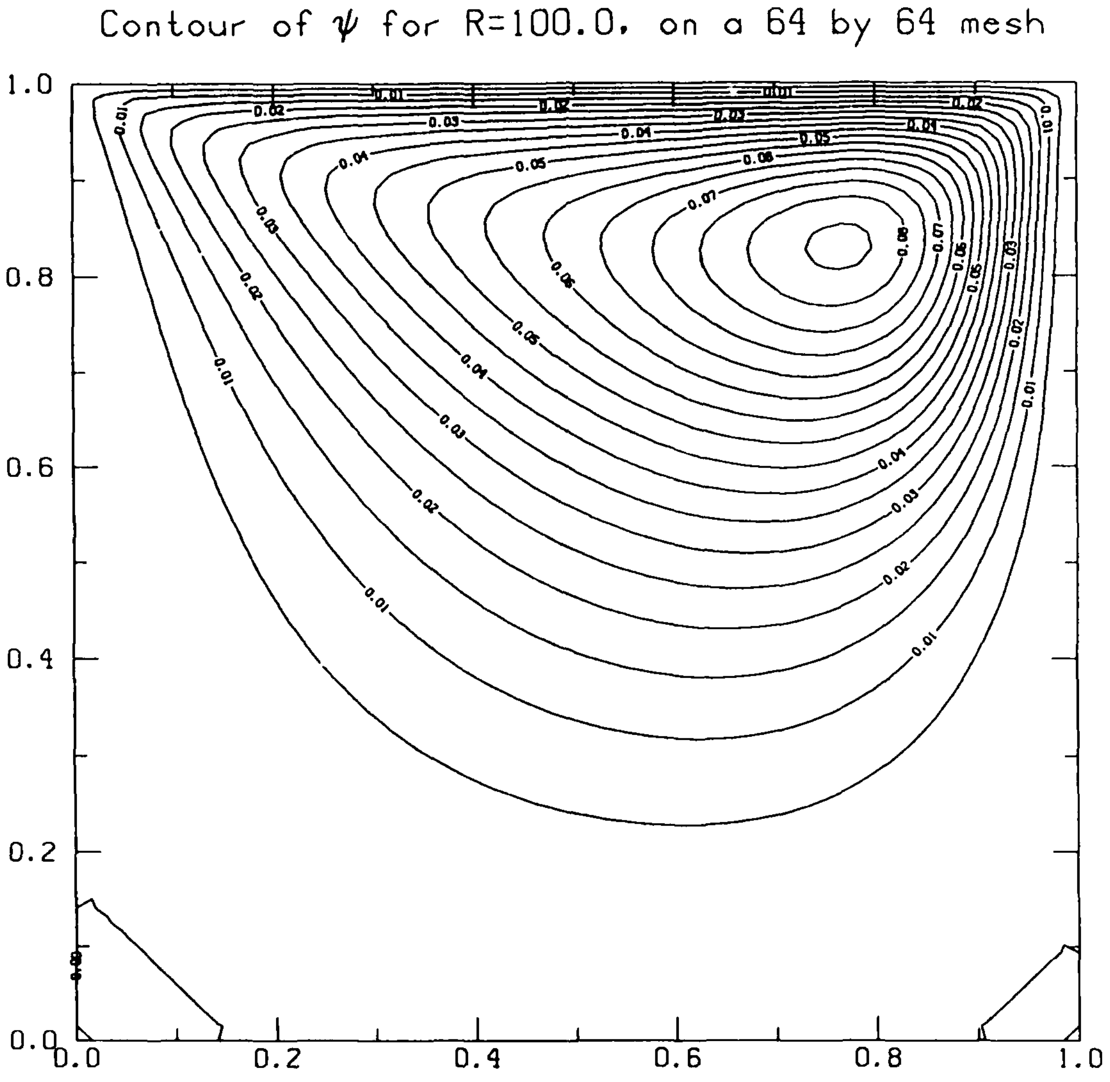


**Figure 1:** Contour of the stream function in the driven cavity problem for Re=100

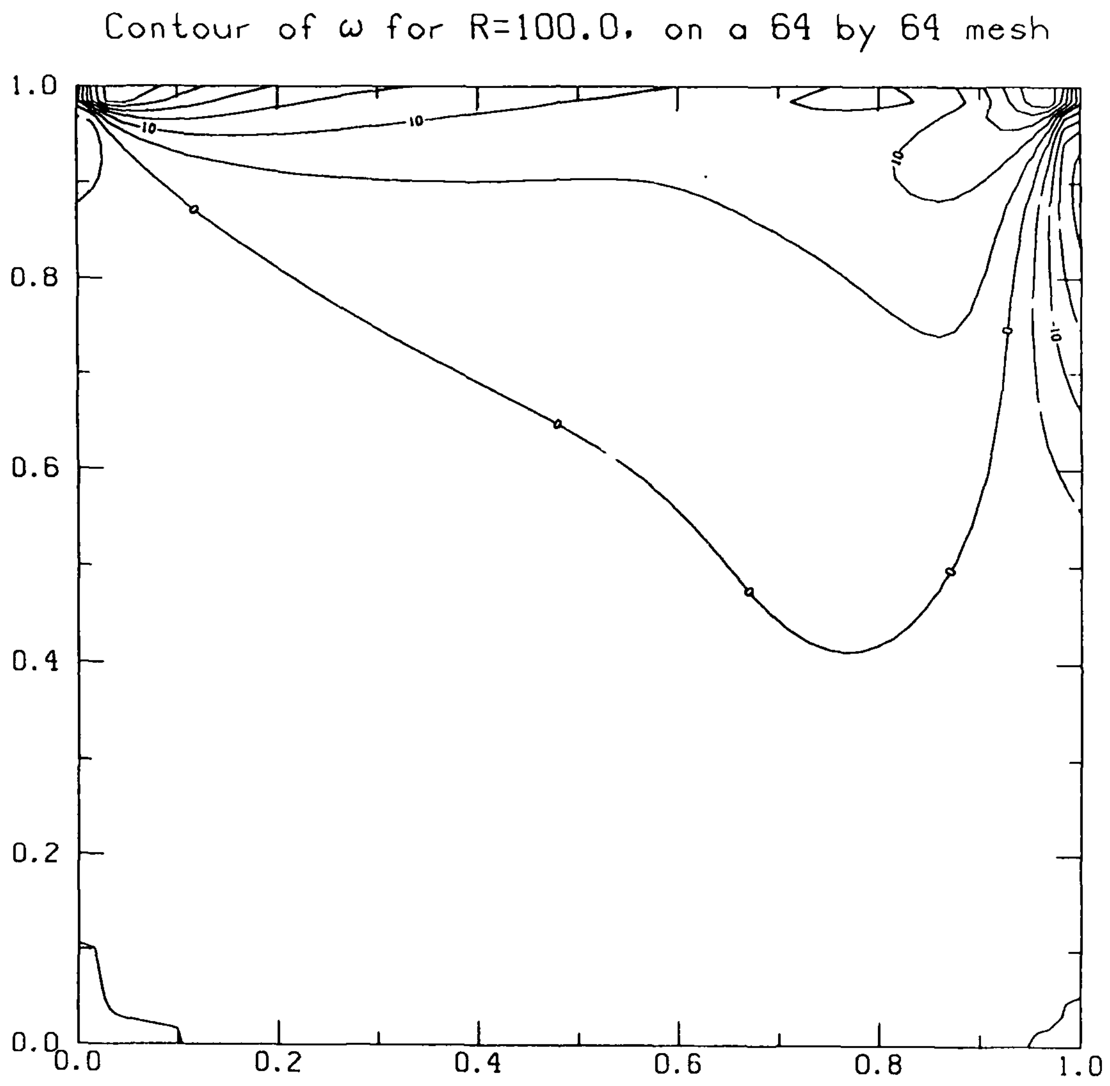


**Figure 2:** Contour of the vorticity in the driven cavity problem for Re=100

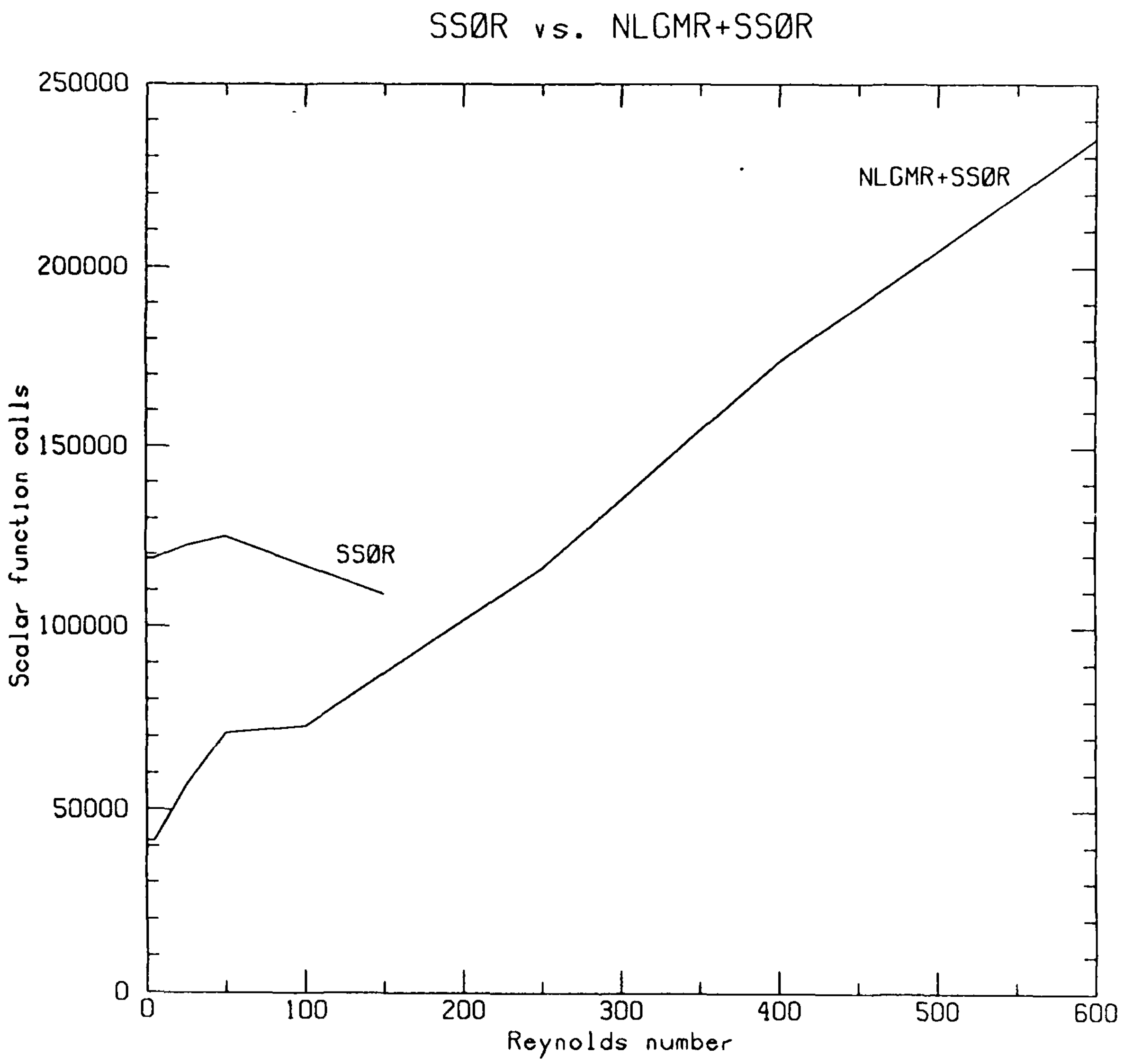
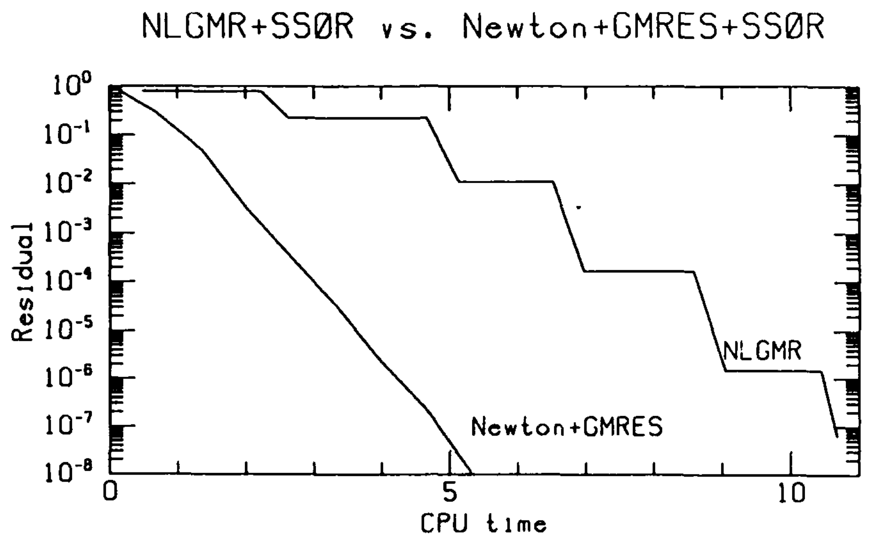


**Figure 3:** Performances of SSOR and NLGMR/SSOR for problem number 1.



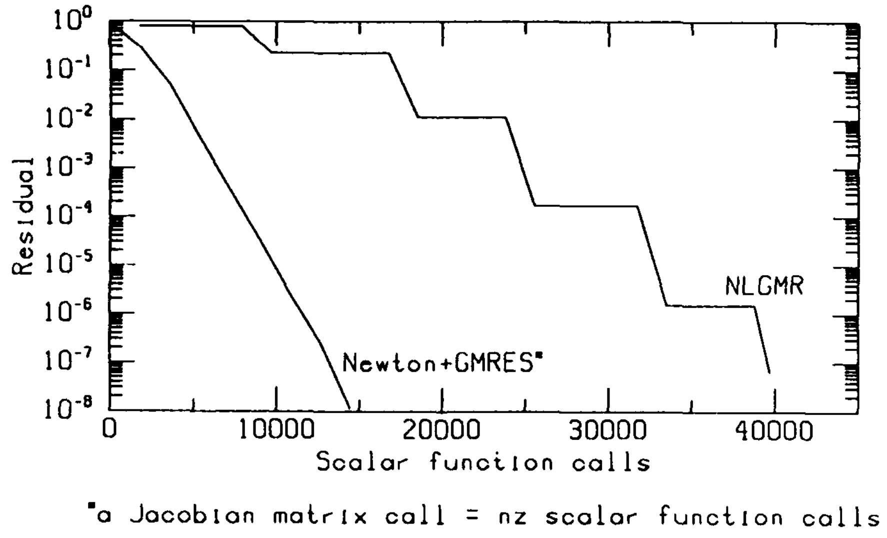


**Figure 4:** Performances of NLGMR/SSOR and Newton/GMRES preconditioned by SSOR for problem number 1.

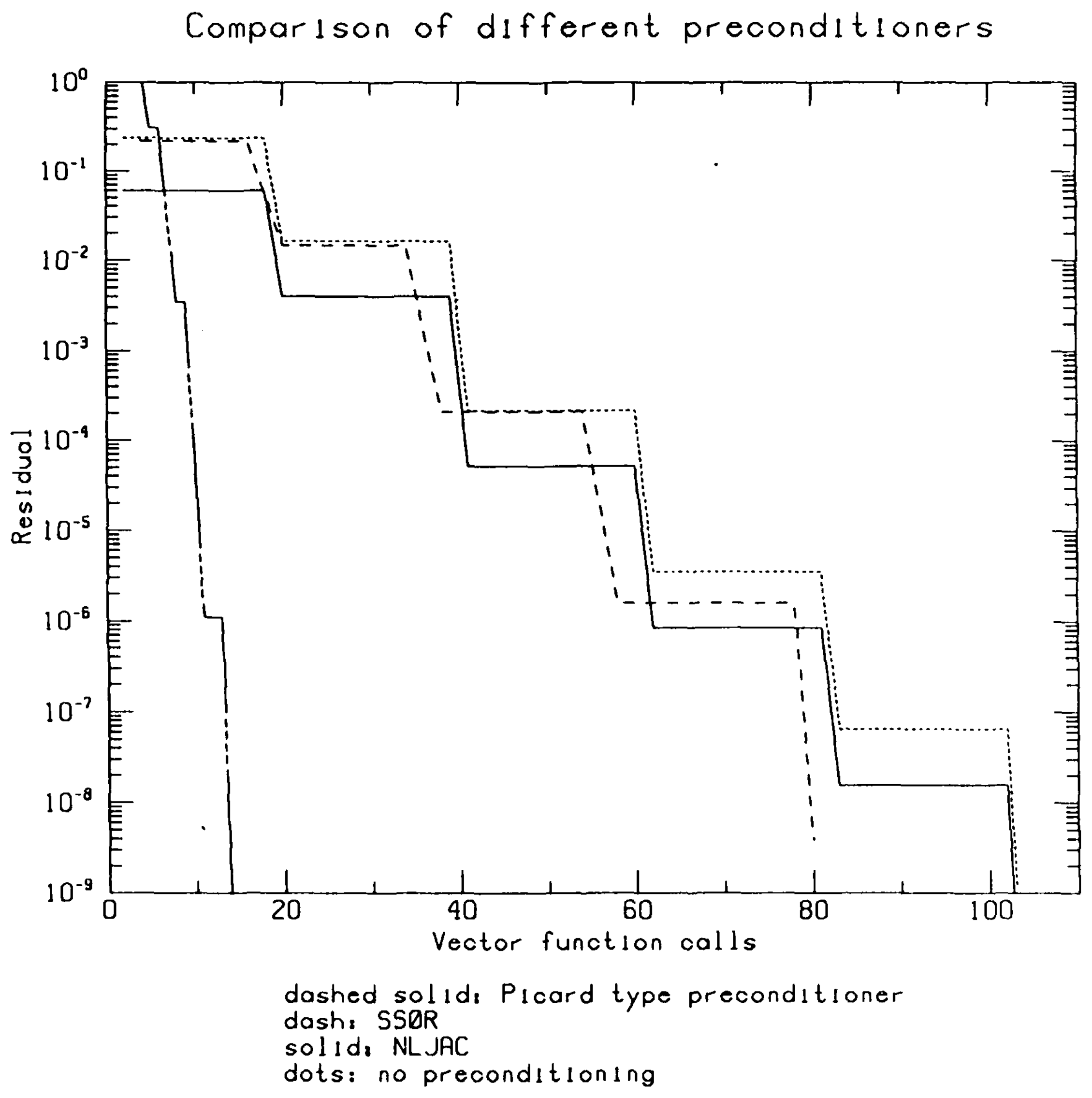


**Figure 5:** Comparison of various preconditioners for problem number 2.

## References

[1] A.S. Benjamin and V.E. Denny, *On the convergence of numerical solutions for the 2-D flows in a cavity at large Re,* Journal of Computational Physics, 33 (1979), pp. 340–358.

[2] P.N. Brown, A.C. Hindmarsh, *Matrix-free methods for stiff systems of ODE'S,* Technical Report UCRL-90770, Laurence Livermore Nat. Lab., 1984.

[3] T.F. Chan and K. R. Jackson, *The use of iterative linear equation solvers in codes for large systems of stiff IVPs for ODEs,* Technical Report 170/84, Univ. of Toronto, 1984.

[4] I.L. Chern, W.L. Miranker, *Dichotomy and conjugate gradients in the stiff initial value problem,* Technical Report 8032-34917, IBM, 1980.

[5] H.C. Elman, *Iterative Methods for Large Sparse Nonsymmetric Systems of Linear Equations,* Ph.D. Thesis, Yale University, Computer Science Dpt., 1982.

[6] L. E. Eriksson and A. Rizzi, Analysis by computer of the convergence of discrete approximations to the Euler equations, *Proceedings of the 1983 AIAA conference, Denver 1983,* AIAA paper number 83-1951, Denver, 1983, pp. 407–442.

[7] W.C. Gear and Y. Saad, *Iterative solution of linear equations in ODE codes,* SIAM J. Sci. Stat. Comp., 4 (1983), pp. 583-601.

[8] U.Ghia, K.N. Ghia, and C.T. Shin, *High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method,* Journal of Computational Physics, 48 (1982), pp. 387–411.

[9] P.E. Gill, W. Murray and M. Wright, *Practical Optimization,* Academic Press, New York, 1981.

[10] A.L. Hageman and D.M. Young, *Applied Iterative Methods,* Academic Press, New York, 1981.

[11] J.J. More, B.S. Garbow, K.E. Hillstrom, *User Guide for MINPACK-1,* Technical Report ANL-80-74, Argonne National Lab., 1980.

[12] J.M. Ortega and W.C. Rheinboldt, *Iterative solution of nonlinear equations in several variables,* Academic Press, New-York, 1970.

[13] S.V. Patankar, D.D. Spadling, *A calculation procedure for heat mass and momentum transfer in three-dimensional parabolic flows,* Intl. J. Heat Mass Transfer, 15 (1972).

[14] Y. Saad, M.H. Schultz, *Conjugate Gradient-like algorithms for solving nonsymmetric linear systems,* Mathematics of Computation, 44/170 (1985), pp. 417–424.

[15] ———, *GMRES: a Generalized Minimal residual algorithm for solving nonsymmetric linear systems,* Technical Report 254, Yale University, 1983. to appear in SISSC.

[16] R. Schreiber and H.B. Keller, *Driven cavity flows by efficient numercial techniques,* Journal of Computational Physics, 49 (1983), pp. 310–333.

[17] J. Stoer and R. Bulirsch, *Introduction to numerical analysis,* Springer Verlag, New York, Heidelberg, 1980.

[18] L.B. Wigton, D.P. Yu, and N.J. Young, GMRES Acceleration of computational Fluid Dynamics Codes, *Proceedings of the 1985 AIAA conference, Denver 1985*, AIAA, Denver, 1985.